1. Tổng quan về Ngôn ngữ lập trình
2. Giới thiệu về lập trình hướng đối tượng
3. Các thành phần cơ bản của ngôn ngữ C
4. Giới thiệu về ngôn ngữ C và môi trường turbo C 3.0

Tổng quan về Ngôn ngữ lập trình
Môn Lập Trình Căn Bản A cung cấp cho sinh viên những kiến thức cơ bản về lập trình thông qua ngôn ngữ lập trình C. Môn học này là nền tảng để tiếp thu hầu hết các môn học khác trong chương trình đào tạo. Mặt khác, nắm vững ngôn ngữ C là cơ sở để phát triển các ứng dụng. Học xong môn này, sinh viên phải nắm được các vấn đề sau: - Khái niệm về ngôn ngữ lập trình. - Khái niệm về kiểu dữ liệu - Kiểu dữ liệu có cấu trúc (cấu trúc dữ liệu). - Khái niệm về giải thuật - Ngôn ngữ biểu diễn giải thuật. - Ngôn ngữ sơ đồ (lưu đồ), sử dụng lưu đồ để biểu diễn các giải thuật. - Tổng quan về Ngôn ngữ lập trình C. - Các kiểu dữ liệu trong C. - Các lệnh có cấu trúc. - Cách thiết kế và sử dụng các hàm trong C. - Một số cấu trúc dữ liệu trong C.

ĐỐI TƯỢNG MÔN HỌC

Môn học lập trình căn bản được dùng để giảng dạy cho các sinh viên sau:

- Sinh viên năm thứ 2 chuyên ngành Tin học, Toán Tin, Lý Tin.

- Sinh viên năm thứ 2 chuyên ngành Điện tử (Viễn thông, Tự động hóa…)

NỘI DUNG CỐT LÕI

Trong khuôn khổ 45 tiết, giáo trình được cấu trúc thành 2 phần: Phần 1 giới thiệu về lập trình cấu trúc, các khái niệm về lập trình, giải thuật… Phần 2 trình bày có hệ thống về ngôn ngữ lập trình C, các câu lệnh, các kiểu dữ liệu…

PHẦN 1: Giới thiệu cấu trúc dữ liệu và giải thuật

PHẦN 2: Giới thiệu về một ngôn ngữ lập trình - Ngôn ngữ lập trình C

Chương 1: Giới thiệu về ngôn ngữ C & môi trường lập trình Turbo C

Chương 2: Các thành phần của ngôn ngữ C

Chương 3: Các kiểu dữ liệu sơ cấp chuẩn và các lệnh đơn

Chương 4: Các lệnh có cấu trúc

Chương 5: Chương trình con

Chương 6: Kiểu mảng

Chương 7: Kiểu con trỏ

Chương 8: Kiểu chuỗi ký tự

Chương 9: Kiểu cấu trúc

Chương 10: Kiểu tập tin

KIẾN THỨC LIÊN QUAN

Để học tốt môn Lập Trình Căn Bản A, sinh viên cần phải có các kiến thức nền tảng sau:

- Kiến thức toán học.

- Kiến thức và kỹ năng thao tác trên máy tính.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] Nguyễn Văn Linh, Giáo trình Tin Học Đại Cương A, Khoa Công Nghệ Thông Tin, Đại học Cần Thơ, 1991.

[2] Nguyễn Đình Tê, Hoàng Đức Hải , Giáo trình lý thuyết và bài tập ngôn ngữ C; Nhà xuất bản Giáo dục, 1999.

[3] Nguyễn Cẩn, C – Tham khảo toàn diện, Nhà xuất bản Đồng Nai, 1996.

[4] Võ Văn Viện, Giúp tự học Lập Trình với ngôn ngữ C, Nhà xuất bản Đồng Nai, 2002.

[5] Brain W. Kernighan & Dennis Ritchie, The C Programming Language, Prentice Hall Publisher, 1988.

TỪ KHÓA

Bài toán, chương trình, giải thuật, ngôn ngữ giả, lưu đồ, biểu thức, gán, rẽ nhánh, lặp, hàm, mảng, con trỏ, cấu trúc, tập tin.

Giới thiệu về lập trình hướng đối tượng
Phần này trình bày về lập trình hướng đối tượng

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (OOP) LÀ GÌ ?

Lập trình hướng đối tượng (Object-Oriented Programming, viết tắt là OOP) là một phương pháp mới trên bước đường tiến hóa của việc lập trình máy tính, nhằm làm cho chương trình trở nên linh hoạt, tin cậy và dễ phát triển. Tuy nhiên để hiểu được OOP là gì, chúng ta hãy bắt đầu từ lịch sử của quá trình lập trình – xem xét OOP đã tiến hóa như thế nào.

Lập trình tuyến tính

Máy tính đầu tiên được lập trình bằng mã nhị phân, sử dụng các công tắt cơ khí để nạp chương trình. Cùng với sự xuất hiện của các thiết bị lưu trữ lớn và bộ nhớ máy tính có dung lượng lớn nên các ngôn ngữ lập trình cấp cao đầu tiên được đưa vào sử dụng . Thay vì phải suy nghĩ trên một dãy các bit và byte, lập trình viên có thể viết một loạt lệnh gần với tiếng Anh và sau đó chương trình dịch thành ngôn ngữ máy.

Các ngôn ngữ lập trình cấp cao đầu tiên được thiết kế để lập các chương trình làm các công việc tương đối đơn giản như tính toán. Các chương trình ban đầu chủ yếu liên quan đến tính toán và không đòi hỏi gì nhiều ở ngôn ngữ lập trình. Hơn nữa phần lớn các chương trình này tương đối ngắn, thường ít hơn 100 dòng.

Khi khả năng của máy tính tăng lên thì khả năng để triển khai các chương trình phức tạp hơn cũng tăng lên. Các ngôn ngữ lập trình ngày trước không còn thích hợp đối với việc lập trình đòi hỏi cao hơn. Các phương tiện cần thiết để sử dụng lại các phần mã chương trình đã viết hầu như không có trong ngôn ngữ lập trình tuyến tính. Thật ra, một đoạn lệnh thường phải được chép lặp lại mỗi khi chúng ta dùng trong nhiều chương trình do đó chương trình dài dòng, logic của chương trình khó hiểu. Chương trình được điều khiển để nhảy đến nhiều chỗ mà thường không có sự giải thích rõ ràng, làm thế nào để chương trình đến chỗ cần thiết hoặc tại sao như vậy.

Ngôn ngữ lập trình tuyến tính không có khả năng kiểm soát phạm vi nhìn thấy của các dữ liệu. Mọi dữ liệu trong chương trình đều là dữ liệu toàn cục nghĩa là chúng có thể bị sửa đổi ở bất kỳ phần nào của chương trình. Việc dò tìm các thay đổi không mong muốn đó của các phần tử dữ liệu trong một dãy mã lệnh dài và vòng vèo đã từng làm cho các lập trình viên rất mất thời gian.

Lập trình cấu trúc

Rõ ràng là các ngôn ngữ mới với các tính năng mới cần phải được phát triển để có thể tạo ra các ứng dụng tinh vi hơn. Vào cuối các năm trong 1960 và 1970, ngôn ngữ lập trình có cấu trúc ra đời. Các chương trình có cấu trúc được tổ chức theo các công việc mà chúng thực hiện.

Về bản chất, chương trình chia nhỏ thành các chương trình con riêng rẽ (còn gọi là hàm hay thủ tục) thực hiện các công việc rời rạc trong quá trình lớn hơn, phức tạp hơn. Các hàm này được giữ càng độc lập với nhau càng nhiều càng tốt, mỗi hàm có dữ liệu và logic riêng.Thông tin được chuyển giao giữa các hàm thông qua các tham số, các hàm có thể có các biến cục bộ mà không một ai nằm bên ngoài phạm vi của hàm lại có thể truy xuất được chúng. Như vậy, các hàm có thể được xem là các chương trình con được đặt chung với nhau để xây dựng nên một ứng dụng.

Mục tiêu là làm sao cho việc triển khai các phần mềm dễ dàng hơn đối với các lập trình viên mà vẫn cải thiện được tính tin cậy và dễ bảo quản chương trình. Một chương trình có cấu trúc được hình thành bằng cách bẻ gãy các chức năng cơ bản của chương trình thành các mảnh nhỏ mà sau đó trở thành các hàm. Bằng cách cô lập các công việc vào trong các hàm, chương trình có cấu trúc có thể làm giảm khả năng của một hàm này ảnh hưởng đến một hàm khác. Việc này cũng làm cho việc tách các vấn đề trở nên dễ dàng hơn. Sự gói gọn này cho phép chúng ta có thể viết các chương trình sáng sủa hơn và giữ được điều khiển trên từng hàm. Các biến toàn cục không còn nữa và được thay thế bằng các tham số và biến cục bộ có phạm vi nhỏ hơn và dễ kiểm soát hơn. Cách tổ chức tốt hơn này nói lên rằng chúng ta có khả năng quản lý logic của cấu

trúc chương trình, làm cho việc triển khai và bảo dưỡng chương trình nhanh hơn và hữu hiện hơn và hiệu quả hơn.

Một khái niệm lớn đã được đưa ra trong lập trình có cấu trúc là sự trừu tượng hóa (Abstraction). Sự trừu tượng hóa có thể xem như khả năng quan sát một sự việc mà không cần xem xét đến các chi tiết bên trong của nó. Trong một chương trình có cấu trúc, chúng ta chỉ cần biết một hàm đã cho có thể làm được một công việc cụ thể gì là đủ. Còn làm thế nào mà công việc đó lại thực hiện được là không quan trọng, chừng nào hàm còn tin cậy được thì còn có thể dùng nó mà không cần phải biết nó thực hiện đúng đắn chức năng của mình như thế nào. Điều này gọi là sự trừu tượng hóa theo chức năng (Functional abstraction) và là nền tảng của lập trình có cấu trúc.

Ngày nay, các kỹ thuật thiết kế và lập trình có cấu trúc được sử rộng rãi. Gần như mọi ngôn ngữ lập trình đều có các phương tiện cần thiết để cho phép lập trình có cấu trúc. Chương trình có cấu trúc dễ viết, dễ bảo dưỡng hơn các chương trình không cấu trúc.

Sự nâng cấp như vậy cho các kiểu dữ liệu trong các ứng dụng mà các lập trình viên đang viết cũng đang tiếp tục diễn ra. Khi độ phức tạp của một chương trình tăng lên, sự phụ thuộc của nó vào các kiểu dữ liệu cơ bản mà nó xử lý cũng tăng theo. Vấn đề trở rõ ràng là cấu trúc dữ liệu trong chương trình quan trọng chẳng kém gì các phép toán thực hiện trên chúng. Điều này càng trở rõ ràng hơn khi kích thước của chương trình càng tăng. Các kiểu dữ liệu được xử lý trong nhiều hàm khác nhau bên trong một chương trình có cấu trúc. Khi có sự thay đổi trong các dữ liệu này thì cũng cần phải thực hiện cả các thay đổi ở mọi nơi có các thao tác tác động trên chúng. Đây có thể là một công việc tốn thời gian và kém hiệu quả đối với các chương trình có hàng ngàn dòng lệnh và hàng trăm hàm trở lên.

Một yếu điểm nữa của việc lập trình có cấu trúc là khi có nhiều lập trình viên làm việc theo nhóm cùng một ứng dụng nào đó. Trong một chương trình có cấu trúc, các lập trình viên được phân công viết một tập hợp các hàm và các kiểu dữ liệu. Vì có nhiều lập trình viên khác nhau quản lý các

hàm riêng, có liên quan đến các kiểu dữ liệu dùng chung nên các thay đổi mà lập trình viên tạo ra trên một phần tử dữ liệu sẽ làm ảnh hưởng đến công việc của tất cả các người còn lại trong nhóm. Mặc dù trong bối cảnh làm việc theo nhóm, việc viết các chương trình có cấu trúc thì dễ dàng hơn nhưng sai sót trong việc trao đổi thông tin giữa các thành viên trong nhóm có thể dẫn tới hậu quả là mất rất nhiều thời gian để sửa chữa chương trình.

Sự trừu tượng hóa dữ liệu

Sự trừu tượng hóa dữ liệu (Data abstraction) tác động trên các dữ liệu cũng tương tự như sự trừu tượng hóa theo chức năng. Khi có trừu tượng hóa dữ liệu, các cấu trúc dữ liệu và các phần tử có thể được sử dụng mà không cần bận tâm đến các chi tiết cụ thể. Chẳng hạn như các số dấu chấm động đã được trừu tượng hóa trong tất cả các ngôn ngữ lập trình, Chúng ta không cần quan tâm cách biểu diễn nhị phân chính xác nào cho số dấu chấm động khi gán một giá trị, cũng không cần biết tính bất thường của phép nhân nhị phân khi nhân các giá trị dấu chấm động. Điều quan trọng là các số dấu chấm động hoạt động đúng đắn và hiểu được.

Sự trừu tượng hóa dữ liệu giúp chúng ta không phải bận tâm về các chi tiết không cần thiết. Nếu lập trình viên phải hiểu biết về tất cả các khía cạnh của vấn đề, ở mọi lúc và về tất cả các hàm của chương trình thì chỉ ít hàm mới được viết ra, may mắn thay trừu tượng hóa theo dữ liệu đã tồn tại sẵn trong mọi ngôn ngữ lập trình đối với các dữ liệu phức tạp như số dấu chấm động. Tuy nhiên chỉ mới gần đây, người ta mới phát triển các ngôn ngữ cho phép chúng ta định nghĩa các kiểu dữ liệu trừu tượng riêng.

Lập trình hướng đối tượng

Khái niệm hướng đối tượng được xây dựng trên nền tảng của khái niệm lập trình có cấu trúc và sự trừu tượng hóa dữ liệu. Sự thay đổi căn bản ở chỗ, một chương trình hướng đối tượng được thiết kế xoay quanh dữ liệu mà chúng ta có thể làm việc trên đó, hơn là theo bản thân chức năng

của chương trình. Điều này hoàn toàn tự nhiên một khi chúng ta hiểu rằng mục tiêu của chương trình là xử lý dữ liệu. Suy cho cùng, công việc mà máy tính thực hiện vẫn thường được gọi là xử lý dữ liệu. Dữ liệu và thao tác liên kết với nhau ở một mức cơ bản (còn có thể gọi là mức thấp), mỗi thứ đều đòi hỏi ở thứ kia có mục tiêu cụ thể, các chương trình hướng đối tượng làm tường minh mối quan hệ này.

Lập trình hướng đối tượng (Object Oriented Programming - gọi tắt là OOP) hay chi tiết hơn là Lập trình định hướng đối tượng, chính là phương pháp lập trình lấy đối tượng làm nền tảng để xây dựng thuật giải, xây dựng chương trình. Thực chất đây không phải là một phương pháp mới mà là một cách nhìn mới trong việc lập trình. Để phân biệt, với phương pháp lập trình theo kiểu cấu trúc mà chúng ta quen thuộc trước đây, hay còn gọi là phương pháp lập trình hướng thủ tục (Procedure-Oriented Programming), người lập trình phân tích một nhiệm vụ lớn thành nhiều công việc nhỏ hơn, sau đó dần dần chi tiết, cụ thể hoá để được các vấn đề đơn giản, để tìm ra cách giải quyết vấn đề dưới dạng những thuật giải cụ thể rõ ràng qua đó dễ dàng minh hoạ bằng ngôn ngữ giải thuật (hay còn gọi các thuật giải này là các chương trình con). Cách thức phân tích và thiết kế như vậy chúng ta gọi là nguyên lý lập trình từ trên xuống (top-down), để thể hiện quá trình suy diễn từ cái chung cho đến cái cụ thể.

Các chương trình con là những chức năng độc lập, sự ghép nối chúng lại với nhau cho chúng ta một hệ thống chương trình để giải quyết vấn đề đặt ra. Chính vì vậy, cách thức phân tích một hệ thống lấy chương trình con làm nền tảng, chương trình con đóng vai trò trung tâm của việc lập trình, được hiểu như phương pháp lập trình hướg về thủ tục. Tuy nhiên, khi phân tích để thiết kế một hệ thống không nhất thiết phải luôn luôn suy nghĩ theo hướng "làm thế nào để giải quyết công việc", chúng ta có thể định hướng tư duy theo phong cách "với một số đối tượng đã có, phải làm gì để giải quyết được công việc đặt ra" hoặc phong phú hơn, "làm cái gì với một số đối tượng đã có đó", từ đó cũng có thể giải quyết được những công việc cụ thể. Với phương pháp phân tích trong đó đối tượng đóng vai trò trùng tâm của việc lập trình như vậy, người ta gọi là nguyên lý lập trình từ dưới lên (Bôttm-up).

Lập trình hướng đối tượng liên kết cấu trúc dữ liệu với các thao tác, theo cách mà tất cả thường nghĩ về thế giới quanh mình. Chúng ta thường gắn một số các hoạt động cụ thể với một loại hoạt động nào đó và đặt các giả thiết của mình trên các quan hệ đó.

Ví dụ1.1: Để dễ hình dùng hơn, chúng ta thủ nhìn qua các công trình xây dựng hiện đại, như sân vận động có mái che hình vòng cung, những kiến trúc thẩm mĩ với đường nét hình cong. Tất cả những sản phẩm đó xuất hiện cùng với những vật liệu xây dựng. Ngày nay, không chỉ chồng lên nhau những viên gạch, những tảng đá để tạo nên những quần thể kiến trúc (như Tháp Chàm Nha Trang, Kim Tự Tháp,...), mà có thể với bêtông, sắt thép và không nhiều lắm những viên gạch, người xây dựng cũng có thể thiết kế những công trình kiến trúc tuyệt mỹ, những toà nhà hiện đại. Chính các chất liệu xây dựng đã làm ảnh hưởng phương pháp xây dựng, chất liệu xây dựng và nguyên lý kết dính caá chất liệu đó lại với nhau cho chúng ta một đối tượng để khảo sát, Chất liệu xây dựng và nguyên lý kết dính các chất liệu lại với nhau được hiểu theo nghĩa dữ liệu và chương trình con tác động trên dữ liệu đó.

Ví dụ1.2: Chúng ta biết rằng một chiếc xe có các bánh xe, di chuyển được và có thể đổi hướng của nó bằng cách quẹo tay lái. Tương tự như thế, một cái cây là một loại thực vật có thân gỗ và lá. Một chiếc xe không phải là một cái cây, mà cái cây không phải là một chiếc xe, chúng ta có thể giả thiết rằng cái mà chúng ta có thể làm được với một chiếc xe thì không thể làm được với một cái cây. Chẳng hạn, thật là vô nghĩa khi muốn lái một cái cây, còn chiếc xe thì lại chẳng lớn thêm được khi chúng ta tưới nước cho nó.

Lập trình hướng đối tượng cho phép chúng ta sử dụng các quá trình suy nghĩ như vậy với các khái niệm trừu tượng được sử dụng trong các chương trình máy tính. Một mẫu tin (record) nhân sự có thể được đọc ra, thay đổi và lưu trữ lại; còn số phức thì có thể được dùng trong các tính toán. Tuy vậy không thể nào lại viết một số phức vào tập tin làm mẫu tin nhân sự và ngược lại hai mẫu tin nhân sự lại không thể cộng với nhau được. Một chương trình hướng đối tượng sẽ xác định đặc điểm và hành vi cụ thể của các kiểu dữ liệu, điều đó cho phép chúng ta biết một cách

chính xác rằng chúng ta có thể có được những gì ở các kiểu dữ liệu khác nhau.

Chúng ta còn có thể tạo ra các quan hệ giữa các kiểu dữ liệu tương tự nhưng khác nhau trong một chương trình hướng đối tượng. Người ta thường tự nhiên phân loại ra mọi thứ, thường đặt mối liên hệ giữa các khái niệm mới với các khái niệm đã có, và thường có thể thực hiện suy diễn giữa chúng trên các quan hệ đó. Hãy quan niệm thế giới theo kiểu cấu trúc cây, với các mức xây dựng chi tiết hơn kế tiếp nhau cho các thế hệ sau so với các thế hệ trước. Đây là phương pháp hiệu quả để tổ chức thế giới quanh chúng ta. Các chương trình hướng đối tượng cũng làm việc theo một phương thức tương tự, trong đó chúng cho phép xây dựng các các cơ cấu dữ liệu và thao tác mới dựa trên các cơ cấu có sẵn, mang theo các tính năng của các cơ cấu nền mà chúng dựa trên đó, trong khi vẫn thêm vào các tính năng mới.

Lập trình hướng đối tượng cho phép chúng ta tổ chức dữ liệu trong chương trình theo một cách tương tự như các nhà sinh học tổ chức các loại thực vật khác nhau. Theo cách nói lập trình đối tượng, xe hơi, cây cối, các số phức, các quyển sách đều được gọi là các lớp (Class).

Một lớp là một bản mẫu mô tả các thông tin cấu trúc dữ liệu, lẫn các thao tác hợp lệ của các phần tử dữ liệu. Khi một phần tử dữ liệu được khai báo là phần tử của một lớp thì nó được gọi là một đối tượng (Object). Các hàm được định nghĩa hợp lệ trong một lớp được gọi là các phương thức (Method) và chúng là các hàm duy nhất có thể xử lý dữ liệu của các đối tượng của lớp đó. Một thực thể (Instance) là một vật thể có thực bên trong bộ nhớ, thực chất đó là một đối tượng (nghĩa là một đối tượng được cấp phát vùng nhớ).

Mỗi một đối tượng có riêng cho mình một bản sao các phần tử dữ liệu của lớp còn gọi là các biến thực thể (Instance variable). Các phương thức định nghĩa trong một lớp có thể được gọi bởi các đối tượng của lớp đó. Điều này được gọi là gửi một thông điệp (Message) cho đối tượng. Các thông điệp này phụ thuộc vào đối tượng, chỉ đối tượng nào nhận thông điệp mới phải làm việc theo thông điệp đó. Các đối tượng đều độc lập

với nhau vì vậy các thay đổi trên các biến thể hiện của đối tượng này không ảnh hưởng gì trên các biến thể hiện của các đối tượng khác và việc gửi thông điệp cho một đối tượng này không ảnh hưởng gì đến các đối tượng khác.

Như vậy, đối tợng được hiểu theo nghĩa là một thực thể mà trong đó caá dữ liệu và thủ tục tác động lên dữ liệu đã được đóng gói lại với nhau. Hay “đối tượng được đặc trưng bởi một số thao tác (operation) và các thông tin (information) ghi nhơ sự tác động của caá thao tác này.”

Ví dụ 1.3: Khi nghiên cứ về ngăn xếp (stack), ngoài các dữ liệu vùng chứa ngăn xếp, đỉnh của ngăn xếp, chúng ta phải cài đặt kèm theo các thao tác như khởi tạo (creat) ngăn xếp, kiểm tra ngăn xếp rỗng (empty), đẩy (push) một phần tử vào ngăn xếp, lấy (pop) một phần tử ra khỏi ngăn xếp. Trên quan điểm lấy đối tượng làm nền tảng, rõ ràng dữ liệu và các thao tác trên dữ liệu luôn gắn bó với nhau, sự kết dính chúng chính là đối tượng chúng ta cần khảo sát.

Các thao tác trong đối tượng được gọi là các phương thức hay hành vi của đối tượng đó. Phương thức và dữ liệu của đối tượng luôn tác động lẫn nhau và có vai trò ngang nhau trong đối tượng, Phương thức của đối tượng được qui định bởi dữ liệu và ngược lại, dữ liệu của đối tượng được đặt trưng bởi các phương thức của đối tượng. Chính nhờ sự gắn bó đó, chúng ta có thể gởi cùng một thông điệp đến những đối tượng khác nhau. Điều này giúp người lập trình không phải xử lý trong chương trình của mình một dãy các cấu trúc điều khiển tuỳ theo thông điệp nhận vào, mà chương trình được xử lý vào thời điểm thực hiện.

Tóm lại, so sánh lập trình cấu trúc với chương trình con làm nền tảng:

Chương trình = Cấu trúc dữ liệu + Thuật giải

Trong lập trình hướng đối tượng chúng ta có:

Đối tượng = Phương thức + Dữ liệu

Đây chính là 2 quan điểm lập trình đang tồn tại và phát triển trong thế giới ngày nay.

MỘT SỐ KHÁI NIỆM MỚI TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Trong phần này, chúng ta tìm hiểu các khái niệm như sự đóng gói, tính kế thừa và tính đa hình. Đây là các khái niệm căn bản, là nền tảng tư tưởng của lập trình hướng đối tượng. Hiểu được khái niệm này, chúng ta bước đầu tiếp cận với phong cách lập trình mới, phong cách lập trình dựa vào đối tượng làm nền tảng mà trong đó quan điểm che dấu thông tin thông qua sư đóng gói là quan điểm trung tâm của vấn đề.

Sự đóng gói (Encapsulation)

Sự đóng gói là cơ chế ràng buộc dữ liệu và thao tác trên dữ liệu đó thành một thể thống nhất, tránh được các tác động bất ngờ từ bên ngoài. Thể thống nhất này gọi là đối tượng.

Trong Objetc Oriented Software Engineering của Ivar Jacibson, tất cả các thông tin của một hệ thống định hướng đối tượng được lưu trữ bên trong đối tượng của nó và chỉ có thể hành động khi các đối tượng đó được ra lệnh thực hiện các thao tác. Như vật, sự đóng gói không chỉ đơn thuần là sự gom chung dữ liệu và chương trình vào trong một khối, chúng còn được hiểu theo nghĩa là sự đồng nhất giữa dữ liệu và các thao tác tác động lên dữ liệu đó.

Trong một đối tượng, dữ liệu hay thao tác hay cả hai có thể là riêng (private) hoặc chung (public) của đối tượng đó. Thao tác hay dữ liệu riêng là thuộc về đối tượng đó chỉ được truy cập bởi các thành phần của đối tượng, điều này nghĩa là thao tác hay dữ liệu riêng không thể truy cập bởi các phần khác của chương trình tồn tại ngoài đối tượng. Khi thao tác hay dữ liệu là chung, các phần khác của chương trình có thể truy cập nó mặc dù nó được định nghĩa trong một đối tượng. Các thành phần chung

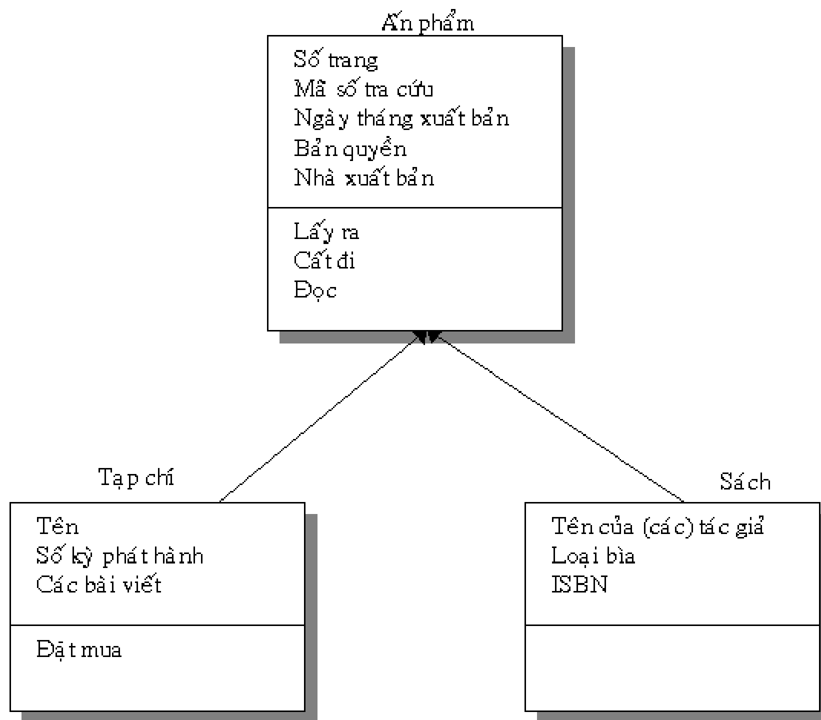
của một đối tượng dùng để cung cấp một giao diện có điều khiển cho các thành thành riêng của đối tượng.

Cơ chế đóng gói là phương thức tốt để thực hiện cơ chế che dấu thông tin so với các ngôn ngữ lập trình cấu trúc.

Tính kế thừa (Inheritance)

Chúng ta có thể xây dựng các lớp mới từ các lớp cũ thông qua sự kế thừa. Một lớp mới còn gọi là lớp dẫn xuất (derived class), có thể thừa hưởng dữ liệu và các phương thức của lớp cơ sở (base class) ban đầu. Trong lớp này, có thể bổ sung các thành phần dữ liệu và các phương thức mới vào những thành phần dữ liệu và các phương thức mà nó thừa hưởng từ lớp cơ sở. Mỗi lớp (kể cả lớp dẫn xuất) có thể có một số lượng bất kỳ các lớp dẫn xuất. Qua cơ cấu kế thừa này, dạng hình cây của các lớp được hình thành. Dạng cây của các lớp trông giống như các cây gia phả vì thế các lớp cơ sở còn được gọi là lớp cha (parent class) và các lớp dẫn xuất được gọi là lớp con (child class).

Ví dụ 1.2: Chúng ta sẽ xây dựng một tập các lớp mô tả cho thư viện các ấn phẩm. Có hai kiểu ấn phẩm: tạp chí và sách. Chúng ta có thể tạo một ấn phẩm tổng quát bằng cách định nghĩa các thành phần dữ liệu tương ứng với số trang, mã số tra cứu, ngày tháng xuất bản, bản quyền và nhà xuất bản. Các ấn phẩm có thể được lấy ra, cất đi và đọc. Đó là các phương thức thực hiện trên một ấn phẩm. Tiếp đó chúng ta định nghĩa hai lớp dẫn xuất tên là tạp chí và sách. Tạp chí có tên, số ký phát hành và chứa nhiều bài của các tác giả khác nhau . Các thành phần dữ liệu tương ứng với các yếu tố này được đặt vào định nghĩa của lớp tạp chí. Tạp chí cũng cần có một phương thức nữa đó là đặt mua. Các thành phần dữ liệu xác định cho sách sẽ bao gồm tên của (các) tác giả, loại bìa (cứng hay mềm) và số hiệu ISBN của nó. Như vậy chúng ta có thể thấy, sách và tạp chí có chung các đặc trưng ấn phẩm, trong khi vẫn có các thuộc tính riêng của chúng.



Hình 1.1: Lớp ấn phẩm và các lớp dẫn xuất của nó.

Với tính kế thừa, chúng ta không phải mất công xây dựng lại từ đầu các lớp mới, chỉ cần bổ sung để có được trong các lớp dẫn xuất các đặc trưng cần thiết.

Tính đa hình (Polymorphism)

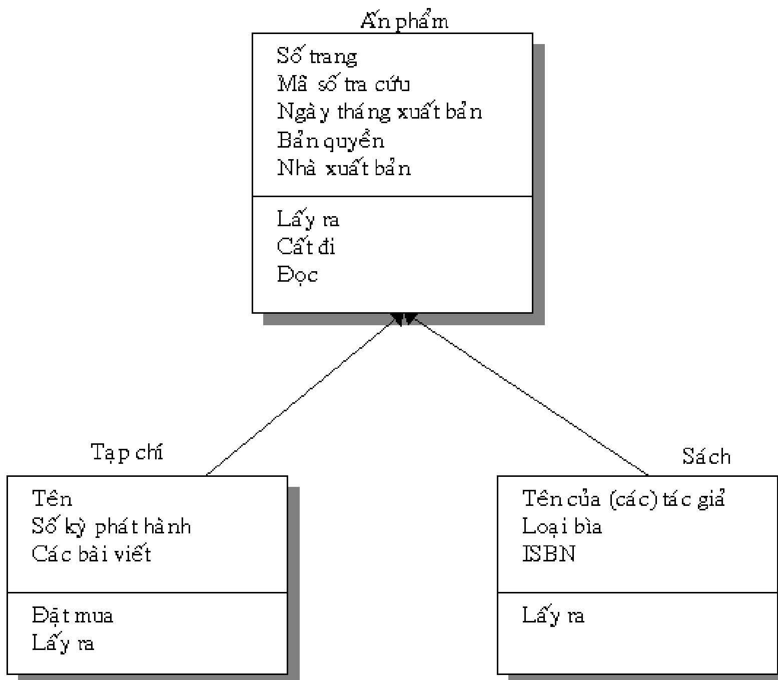
Đó là khả năng để cho một thông điệp có thể thay đổi cách thực hiện của nó theo lớp cụ thể của đối tượng nhận thông điệp. Khi một lớp dẫn xuất được tạo ra, nó có thể thay đổi cách thực hiện các phương thức nào đó mà nó thừa hưởng từ lớp cơ sở của nó. Một thông điệp khi được gởi đến một đối tượng của lớp cơ sở, sẽ dùng phương thức đã định nghĩa cho nó trong lớp cơ sở. Nếu một lớp dẫn xuất định nghĩa lại một phương thức thừa hưởng từ lớp cơ sở của nó thì một thông điệp có cùng tên với phương thức này, khi được gởi tới một đối tượng của lớp dẫn xuất sẽ gọi phương thức đã định nghĩa cho lớp dẫn xuất.

Như vậy đa hình là khả năng cho phép gởi cùng một thông điệp đến những đối tượng khác nhau có cùng chung một đặc điểm, nói cách khác

thông điệp được gởi đi không cần biết thực thể nhận thuộc lớp nào, chỉ biết rằng tập hợp các thực thể nhận có chung một tính chất nào đó. Chẳng hạn, thông điệp “vẽ hình” được gởi đến cả hai đối tượng hình hộp và hình tròn. Trong hai đối tượng này đều có chung phương thức vẽ hình, tuy nhiên tuỳ theo thời điểm mà đối tượng nhận thông điệp, hình tương ứng sẽ được vẽ lên.

Trong các ngôn ngữ lập trình OOP, tính đa hình thể hiện qua khả năng cho phép mô tả những phương thức có tên giống nhau trong các lớp khác nhau. Đặc điểm này giúp người lập trình không phải viết những cấu trúc điều khiển rườm rà trong chương trình, các khả năng khác nhau của thông điệp chỉ thực sự đòi hỏi khi chương trình thực hiện.

Ví dụ 1.3: Xét lại ví dụ 1.2, chúng ta thấy rằng cả tạp chí và và sách đều phải có khả năng lấy ra. Tuy nhiên phương pháp lấy ra cho tạp chí có khác so với phương pháp lấy ra cho sách, mặc dù kết quả cuối cùng giống nhau. Khi phải lấy ra tạp chí, thì phải sử dụng phương pháp lấy ra riêng cho tạp chí (dựa trên một bản tra cứu) nhưng khi lấy ra sách thì lại phải sử dụng phương pháp lấy ra riêng cho sách (dựa trên hệ thống phiếu lưu trữ). Tính đa hình cho phép chúng ta xác định một phương thức để lấy ra một tạp chí hay một cuốn sách. Khi lấy ra một tạp chí nó sẽ dùng phương thức lấy ra dành riêng cho tạp chí, còn khi lấy ra một cuốn sách thì nó sử dụng phương thức lấy ra tương ứng với sách. Kết quả là chỉ cần một tên phương thức duy nhất được dùng cho cả hai công việc tiến hành trên hai lớp dẫn xuất có liên quan, mặc dù việc thực hiện của phương thức đó thay đổi tùy theo từng lớp.



Tính đa hình dựa trên sự nối kết (Binding), đó là quá trình gắn một phương thức với một hàm thực sự. Khi các phương thức kiểu đa hình được sử dụng thì trình biên dịch chưa thể xác định hàm nào tương ứng với phương thức nào sẽ được gọi. Hàm cụ thể được gọi sẽ tuỳ thuộc vào việc phần tử nhận thông điệp lúc đó là thuộc lớp nào, do đó hàm được gọi chỉ xác định được vào lúc chương trình chạy. Điều này gọi là sự kết nối muộn (Late binding) hay kết nối lúc chạy (Runtime binding) vì nó xảy ra khi chương trình đang thực hiện.

Hình 1.2: Minh họa tính đa hình đối với lớp ấn phẩm và các lớp dẫn xuất của nó.

CÁC NGÔN NGỮ VÀ VÀI ỨNG DỤNG CỦA OOP

Xuất phát từ tư tưởng của ngôn ngữ SIMULA67, trung tâm nghiên cứu Palo Alto (PARC) của hãng XEROR đã tập trung 10 năm nghiên cứu để hoàn thiện ngôn ngữ OOP đầu tiên với tên gọi là Smalltalk. Sau đó các ngôn ngữ OOP lần lượt ra đời như Eiffel, Clos, Loops, Flavors, Object Pascal, Object C, C++, Delphi, Java…

Chính XEROR trên cơ sở ngôn ngữ OOP đã đề ra tư tưởng giao diện biểu tượng trên màn hình (icon base screen interface), kể từ đó Apple Macintosh cũng như Microsoft Windows phát triển giao diện đồ họa như ngày nay. Trong Microsoft Windows, tư tưởng OOP được thể hiện một cách rõ nét nhất đó là "chúng ta click vào đối tượng", mỗi đối tượng có thể là control menu, control menu box, menu bar, scroll bar, button, minimize box, maximize box, … sẽ đáp ứng công việc tùy theo đặc tính của đối tượng. Turbo Vision của hãng Borland là một ứng dụng OOP tuyệt vời, giúp lập trình viên không quan tâm đến chi tiết của chương trình gia diện mà chỉ cần thực hiện các nội dung chính của vấn đề.

Các thành phần cơ bản của ngôn ngữ C
Học xong chương này, sinh viên sẽ nắm được các vấn đề sau: - Bộ chữ viết trong C. - Các từ khóa. - Danh biểu. - Các kiểu dữ liệu - Biến và các biểu thức trong C. - Cấu trúc của một chương trình viết bằng ngôn ngữ lập trình C

Bộ chữ viết trong C

Bộ chữ viết trong ngôn ngữ C bao gồm những ký tự, ký hiệu sau: (phân biệt chữ in hoa và in thường):

- 26 chữ cái latinh lớn A,B,C...Z
- 26 chữ cái latinh nhỏ a,b,c ...z.
- 10 chữ số thập phân 0,1,2...9.
- Các ký hiệu toán học: +, -, *, /, =, <, >, (,)
- Các ký hiệu đặc biệt: :. , ; " ' _ @ # $! ^ [ ] { } ...
- Dấu cách hay khoảng trống.

Các từ khoá trong C

Từ khóa là các từ dành riêng (reserved words) của C mà người lập trình có thể sử dụng nó trong chương trình tùy theo ý nghĩa của từng từ. Ta không được dùng từ khóa để đặt cho các tên của riêng mình. Các từ khóa của Turbo C 3.0 bao gồm:

asm auto break case cdecl char

class const continue _cs default delete

do double _ds else enum _es

extern _export far _fastcall float for

friend goto huge if inline int

interrupt _loadds long near new operator

pascal private protected public register return

_saveregs _seg short signed sizeof _ss

static struct switch template this typedef

union unsigned virtual void volatile while

Cặp dấu ghi chú thích

Khi viết chương trình đôi lúc ta cần phải có vài lời ghi chú về 1 đoạn chương trình nào đó để dễ nhớ và dễ điều chỉnh sau này; nhất là phần nội dung ghi chú phải không thuộc về chương trình (khi biên dịch phần này bị bỏ qua). Trong ngôn ngữ lập trình C, nội dung chú thích phải được viết trong cặp dấu /* và */.

Ví dụ :

```
#include <stdio.h>

#include<conio.h>

int main ()

{

char ten[50]; /* khai bao bien ten kieu char 50 ky tu */

/*Xuat chuoi ra man hinh*/

printf(“Xin cho biet ten cua ban !”);

scanf(“%s”,ten); /*Doc vao 1 chuoi la ten cua ban*/

printf(“Xin chao ban %s\n ”,ten);

printf(“Chao mung ban den voi Ngon ngu lap trinh C”);
```

```
/*Dung chuong trinh, cho go phim*/

getch();

return 0;

}
```

CÁC KIỂU DỮ LIỆU SƠ CẤP CHUẨN TRONG C

Các kiểu dữ liệu sơ cấp chuẩn trong C có thể được chia làm 2 dạng : kiểu số nguyên, kiểu số thực.

Kiểu số nguyên

Kiểu số nguyên là kiểu dữ liệu dùng để lưu các giá trị nguyên hay còn gọi là kiểu đếm được. Kiểu số nguyên trong C được chia thành các kiểu dữ liệu con, mỗi kiểu có một miền giá trị khác nhau

Kiểu số nguyên 1 byte (8 bits)

Kiểu số nguyên một byte gồm có 2 kiểu sau:

STT	Kiểu dữ liệu	Miền giá trị (Domain)
1	unsigned char	Từ 0 đến 255 (tương đương 256 ký tự trong bảng mã ASCII)

2	char	Từ -128 đến 127

Kiểu unsigned char: lưu các số nguyên dương từ 0 đến 255.

=> Để khai báo một biến là kiểu ký tự thì ta khai báo biến kiểu unsigned char. Mỗi số trong miền giá trị của kiểu unsigned char tương ứng với một ký tự trong bảng mã ASCII .

Kiểu char: lưu các số nguyên từ -128 đến 127. Kiểu char sử dụng bit trái nhất để làm bit dấu.

=> Nếu gán giá trị > 127 cho biến kiểu char thì giá trị của biến này có thể là số âm (?).

Kiểu số nguyên 2 bytes (16 bits)

Kiểu số nguyên 2 bytes gồm có 4 kiểu sau:

STT	Kiểu dữ liệu	Miền giá trị (Domain)
1	enum	Từ -32,768 đến 32,767
2	unsigned int	Từ 0 đến 65,535
3	short int	Từ -32,768 đến 32,767
4	int	Từ -32,768 đến 32,767

Kiểu enum, short int, int : Lưu các số nguyên từ -32768 đến 32767. Sử dụng bit bên trái nhất để làm bit dấu.

=> Nếu gán giá trị >32767 cho biến có 1 trong 3 kiểu trên thì giá trị của biến này có thể là số âm.

Kiểu unsigned int: Kiểu unsigned int lưu các số nguyên dương từ 0 đến 65535.

Kiểu số nguyên 4 byte (32 bits)

Kiểu số nguyên 4 bytes hay còn gọi là số nguyên dài (long) gồm có 2 kiểu sau:

STT	Kiểu dữ liệu	Miền giá trị (Domain)
1	unsigned long	Từ 0 đến 4,294,967,295
2	long	Từ -2,147,483,648 đến 2,147,483,647

Kiểu long : Lưu các số nguyên từ -2147483658 đến 2147483647. Sử dụng bit bên trái nhất để làm bit dấu.

=> Nếu gán giá trị >2147483647 cho biến có kiểu long thì giá trị của biến này có thể là số âm.

Kiểu unsigned long: Kiểu unsigned long lưu các số nguyên dương từ 0 đến 4294967295

Kiểu số thực

Kiểu số thực dùng để lưu các số thực hay các số có dấu chấm thập phân gồm có 3 kiểu sau:

STT	Kiểu dữ liệu	Kích thước (Size)	Miền giá trị (Domain)
1	float	4 bytes	Từ 3.4 * 10-38 đến 3.4 * 1038
2	double	8 bytes	Từ 1.7 * 10-308 đến 1.7 * 10308
3	long double	10 bytes	Từ 3.4 *10-4932 đến 1.1 *104932

Mỗi kiểu số thực ở trên đều có miền giá trị và độ chính xác (số số lẻ) khác nhau. Tùy vào nhu cầu sử dụng mà ta có thể khai báo biến thuộc 1 trong 3 kiểu trên.

Ngoài ra ta còn có kiểu dữ liệu void, kiểu này mang ý nghĩa là kiểu rỗng không chứa giá trị gì cả.

Tên và hằng trong C

Tên (danh biểu)

Tên hay còn gọi là danh biểu (identifier) được dùng để đặt cho chương trình, hằng, kiểu, biến, chương trình con... Tên có hai loại là tên chuẩn và tên do người lập trình đặt.

Tên chuẩn là tên do C đặt sẵn như tên kiểu: int, char, float,…; tên hàm: sin, cos...

Tên do người lập trình tự đặt để dùng trong chương trình của mình. Sử dụng bộ chữ cái, chữ số và dấu gạch dưới (_) để đặt tên, nhưng phải tuân thủ quy tắc:

- Bắt đầu bằng một chữ cái hoặc dấu gạch dưới.
- Không có khoảng trống ở giữa tên.
- Không được trùng với từ khóa.
- Độ dài tối đa của tên là không giới hạn, tuy nhiên chỉ có 31 ký tự đầu tiên là có ý nghĩa.
- Không cấm việc đặt tên trùng với tên chuẩn nhưng khi đó ý nghĩa của tên chuẩn không còn giá trị nữa.

Ví dụ: tên do người lập trình đặt: Chieu_dai, Chieu_Rong, Chu_Vi, Dien_Tich

Tên không hợp lệ: Do Dai, 12A2,…

Hằng (Constant)

Là đại lượng không đổi trong suốt quá trình thực thi của chương trình.

Hằng có thể là một chuỗi ký tự, một ký tự, một con số xác định. Chúng có thể được biểu diễn hay định dạng (Format) với nhiều dạng thức khác nhau.

Hằng số thực

Số thực bao gồm các giá trị kiểu float, double, long double được thể hiện theo 2 cách sau:

- Cách 1: Sử dụng cách viết thông thường mà chúng ta đã sử dụng trong các môn Toán, Lý, …Điều cần lưu ý là sử dụng dấu thập phân là dấu chấm (.);

Ví dụ: 123.34-223.3333.00-56.0

- Cách 2: Sử dụng cách viết theo số mũ hay số khoa học. Một số thực được tách làm 2 phần, cách nhau bằng ký tự e hay E

Phần giá trị: là một số nguyên hay số thực được viết theo cách 1.

Phần mũ: là một số nguyên

Giá trị của số thực là: Phần giá trị nhân với 10 mũ phần mũ.

Ví dụ: 1234.56e-3= 1.23456 (là số 1234.56 * 10-3)

-123.45E4 = -1234500 (là -123.45 *104)

Hằng số nguyên

Số nguyên gồm các kiểu int (2 bytes) , long (4 bytes) được thể hiện theo những cách sau.

- Hằng số nguyên 2 bytes (int) hệ thập phân: Là kiểu số mà chúng ta sử dụng thông thường, hệ thập phân sử dụng các ký số từ 0 đến 9 để biểu diễn một giá trị nguyên.

Ví dụ: 123 (một trăm hai mươi ba), -242 (trừ hai trăm bốn mươi hai).

- Hằng số nguyên 2 byte (int) hệ bát phân: Là kiểu số nguyên sử dụng 8 ký số từ 0 đến 7 để biểu diễn một số nguyên.

Cách biểu diễn: 0<các ký số từ 0 đến 7>

Ví dụ : 0345 (số 345 trong hệ bát phân)

-020 (số -20 trong hệ bát phân)

Cách tính giá trị thập phân của số bát phân như sau:

Số bát phân : 0dndn-1dn-2…d1d0 (di có giá trị từ 0 đến 7)

=> Giá trị thập phân= $\sum_{i=0}^{n} d_i * 8^i$

0345=229 , 020=16

- Hằng số nguyên 2 byte (int) hệ thập lục phân: Là kiểu số nguyên sử dụng 10 ký số từ 0 đến 9 và 6 ký tự A, B, C, D, E ,F để biểu diễn một số nguyên.

Ký tự giá trị

A10

B11

C12

D13

E14

F15

Cách biểu diễn: 0x<các ký số từ 0 đến 9 và 6 ký tự từ A đến F>

Ví dụ:

0x345 (số 345 trong hệ 16)

0x20 (số 20 trong hệ 16)

0x2A9 (số 2A9 trong hệ 16)

Cách tính giá trị thập phân của số thập lục phân như sau:

Số thập lục phân : 0xdndn-1dn-2…d1d0 (di từ 0 đến 9 hoặc A đến F)

=> Giá trị thập phân= $\sum_{i=0}^{n} d_i * 16^i$

0x345=827 , 0x20=32 , 0x2A9= 681

- Hằng số nguyên 4 byte (long): Số long (số nguyên dài) được biểu diễn như số int trong hệ thập phân và kèm theo ký tự l hoặc L. Một số nguyên nằm ngoài miền giá trị của số int (2 bytes) là số long (4 bytes).

Ví dụ: 45345L hay 45345l hay 45345

- Các hằng số còn lại: Viết như cách viết thông thường (không có dấu phân cách giữa 3 số)

Ví dụ:

12 (mười hai)

12.45 (mười hai chấm 45)

1345.67 (một ba trăm bốn mươi lăm chấm sáu mươi bảy)

Hằng ký tự

Hằng ký tự là một ký tự riêng biệt được viết trong cặp dấu nháy đơn (‘). Mỗi một ký tự tương ứng với một giá trị trong bảng mã ASCII. Hằng ký tự cũng được xem như trị số nguyên.

Ví dụ: ‘a’, ‘A’, ‘0’, ‘9’

Chúng ta có thể thực hiện các phép toán số học trên 2 ký tự (thực chất là thực hiện phép toán trên giá trị ASCII của chúng)

Hằng chuỗi ký tự

Hằng chuỗi ký tự là một chuỗi hay một xâu ký tự được đặt trong cặp dấu nháy kép (“).

Ví dụ: “Ngon ngu lap trinh C”, “Khoa CNTT-DHCT”, “NVLinh-DVHieu”

Chú ý:

1. Một chuỗi không có nội dung “” được gọi là chuỗi rỗng.

2. Khi lưu trữ trong bộ nhớ, một chuỗi được kết thúc bằng ký tự NULL (‘\0’: mã Ascii là 0).

3. Để biểu diễn ký tự đặc biệt bên trong chuỗi ta phải thêm dấu \ phía trước.

Ví dụ: “I’m a student” phải viết “I\’m a student”

“Day la ky tu “dac biet”” phải viết “Day la ky tu \”dac biet\”“

BIẾN VÀ BIỂU THỨC

Biến

Biến là một đại lượng được người lập trình định nghĩa và được đặt tên thông qua việc khai báo biến. Biến dùng để chứa dữ liệu trong quá trình thực hiện chương trình và giá trị của biến có thể bị thay đổi trong quá trình này. Cách đặt tên biến giống như cách đặt tên đã nói trong phần trên.

Mỗi biến thuộc về một kiểu dữ liệu xác định và có giá trị thuộc kiểu đó.

Cú pháp khai báo biến:

<Kiểu dữ liệu> Danh sách các tên biến cách nhau bởi dấu phẩy;

Ví dụ:

int a, b, c; /*Ba biến a, b,c có kiểu int*/

long int chu_vi; /*Biến chu_vi có kiểu long*/

float nua_chu_vi; /*Biến nua_chu_vi có kiểu float*/

double dien_tich; /*Biến dien_tich có kiểu double*/

Lưu ý: Để kết thúc 1 lệnh phải có dấu chấm phẩy (;) ở cuối lệnh.

Vị trí khai báo biến trong C

Trong ngôn ngữ lập trình C, ta phải khai báo biến đúng vị trí. Nếu khai báo (đặt các biến) không đúng vị trí sẽ dẫn đến những sai sót ngoài ý muốn mà người lập trình không lường trước (hiệu ứng lề). Chúng ta có 2 cách đặt vị trí của biến như sau:

a) Khai báo biến ngoài: Các biến này được đặt bên ngoài tất cả các hàm và nó có tác dụng hay ảnh hưởng đến toàn bộ chương trình (còn gọi là biến toàn cục).

Ví dụ:

int i; /*Bien ben ngoai */

float pi; /*Bien ben ngoai*/

int main()

{ … }

b) Khai báo biến trong: Các biến được đặt ở bên trong hàm, chương trình chính hay một khối lệnh. Các biến này chỉ có tác dụng hay ảnh hưởng đến hàm, chương trình hay khối lệnh chứa nó. Khi khai báo biến, phải đặt các biến này ở đầu của khối lệnh, trước các lệnh gán, …

Ví dụ 1:

```
#include <stdio.h>

#include<conio.h>

int bienngoai;/*khai bao bien ngoai*/

int main ()

{ int j,i;/*khai bao bien ben trong chuong trinh chinh*/

clrscr();

i=1; j=2;

bienngoai=3;

printf("\n Gia7 tri cua i la %d",i);

/*%d là số nguyên, sẽ biết sau */

printf("\n Gia tri cua j la %d",j);

printf("\n Gia tri cua bienngoai la %d",bienngoai);

getch();

return 0;

}
```

Ví dụ 2:

```
#include <stdio.h>

#include<conio.h>

int main ()

{ int i, j;/*Bien ben trong*/
```

```
clrscr();

i=4; j=5;

printf("\n Gia tri cua i la %d",i);

printf("\n Gia tri cua j la %d",j);

if(j>i)

{

int hieu=j-i; /*Bien ben trong */

printf("\n Hieu so cua j tru i la %d",hieu);

}

else

{

int hieu=i-j; /*Bien ben trong*/

printf("\n Gia tri cua i tru j la %d",hieu);

}

getch();

return 0;

}
```

Biểu thức

Biểu thức là một sự kết hợp giữa các toán tử (operator) và các toán hạng (operand) theo đúng một trật tự nhất định.

Mỗi toán hạng có thể là một hằng, một biến hoặc một biểu thức khác.

Trong trường hợp, biểu thức có nhiều toán tử, ta dùng cặp dấu ngoặc đơn () để chỉ định toán tử nào được thực hiện trước.

Ví dụ: Biểu thức nghiệm của phương trình bậc hai:

(-b + sqrt(Delta))/(2*a)

Trong đó 2 là hằng; a, b, Delta là biến.

Các toán tử số học

Trong ngôn ngữ C, các toán tử +, -, *, / làm việc tương tự như khi chúng làm việc trong các ngôn ngữ khác. Ta có thể áp dụng chúng cho đa số kiểu dữ liệu có sẵn được cho phép bởi C. Khi ta áp dụng phép / cho một số nguyên hay một ký tự, bất kỳ phần dư nào cũng bị cắt bỏ. Chẳng hạn, 5/2 bằng 2 trong phép chia nguyên.

Toán tử	Ý nghĩa
+	Cộng
-	Trừ
*	Nhân
/	Chia
%	Chia lấy phần dư

--	Giảm 1 đơn vị
++	Tăng 1 đơn vị

Tăng và giảm (++ & --)

Toán tử ++ thêm 1 vào toán hạng của nó và – trừ bớt 1. Nói cách khác:

x = x + 1 giống như ++x

x = x – 1 giống như x—

Cả 2 toán tử tăng và giảm đều có thể tiền tố (đặt trước) hay hậu tố (đặt sau) toán hạng. Ví dụ:x = x + 1 có thể viết x++ (hay ++x)

Tuy nhiên giữa tiền tố và hậu tố có sự khác biệt khi sử dụng trong 1 biểu thức. Khi 1 toán tử tăng hay giảm đứng trước toán hạng của nó, C thực hiện việc tăng hay giảm trước khi lấy giá trị dùng trong biểu thức. Nếu toán tử đi sau toán hạng, C lấy giá trị toán hạng trước khi tăng hay giảm nó. Tóm lại:

x = 10

y = ++x //y = 11

Tuy nhiên:

x = 10

x = x++ //y = 10

Thứ tự ưu tiên của các toán tử số học:

++ -- sau đó là * / % rồi mới đến + -

Các toán tử quan hệ và các toán tử Logic

Ý tưởng chính của toán tử quan hệ và toán tử Logic là đúng hoặc sai. Trong C mọi giá trị khác 0 được gọi là đúng, còn sai là 0. Các biểu thức sử dụng các toán tử quan hệ và Logic trả về 0 nếu sai và trả về 1 nếu đúng.

Toán tử	Ý nghĩa
Các toán tử quan hệ	
>	Lớn hơn
>=	Lớn hơn hoặc bằng
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
==	Bằng
!=	Khác
Các toán tử Logic	
&&	AND
||	OR
!	NOT

Bảng chân trị cho các toán tử Logic:

<table>
<tr><th>P</th><th>q</th><th>p&&q</th><th>p||q</th><th>!p</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td></tr>
</table>

Các toán tử quan hệ và Logic đều có độ ưu tiên thấp hơn các toán tử số học. Do đó một biểu thức như: 10 > 1+ 12 sẽ được xem là 10 > (1 + 12) và kết quả là sai (0).

Ta có thể kết hợp vài toán tử lại với nhau thành biểu thức như sau:

10>5&&!(10<9)||3<=4 Kết quả là đúng

Thứ tự ưu tiên của các toán tử quan hệ là Logic

Cao nhất:!

>>=<<=

==!=

&&

Thấp nhất:||

Các toán tử Bitwise:

Các toán tử Bitwise ý nói đến kiểm tra, gán hay sự thay đổi các Bit thật sự trong 1 Byte của Word, mà trong C chuẩn là các kiểu dữ liệu và biến

char, int. Ta không thể sử dụng các toán tử Bitwise với dữ liệu thuộc các kiểu float, double, long double, void hay các kiểu phức tạp khác.

Toán tử	Ý nghĩa
&	AND
|	OR
^	XOR
~	NOT
>>	Dịch phải
<<	Dịch trái

Bảng chân trị của toán tử ^ (XOR)

p	q	p^q
0	0	0
0	1	1
1	0	1

1	1	0

Toán tử ? cùng với :

C có một toán tử rất mạnh và thích hợp để thay thế cho các câu lệnh của If-Then-Else. Cú pháp của việc sử dụng toán tử ? là:

E1?E2:E3

Trong đó E1, E2, E3 là các biểu thức.

Ý nghĩa: Trước tiên E1 được ước lượng, nếu đúng E2 được ước lượng và nó trở thành giá trị của biểu thức; nếu E1 sai, E2 được ước lượng và trở thành giá trị của biểu thức.

Ví dụ:

X = 10

Y = X > 9 ? 100 : 200

Thì Y được gán giá trị 100, nếu X nhỏ hơn 9 thì Y sẽ nhận giá trị là 200. Đoạn mã này tương đương cấu trúc if như sau:

X = 10

if (X < 9) Y = 100

else Y = 200

Toán tử con trỏ & và *

Một con trỏ là địa chỉ trong bộ nhớ của một biến. Một biến con trỏ là một biến được khai báo riêng để chứa một con trỏ đến một đối tượng của kiểu đã chỉ ra nó. Ta sẽ tìm hiểu kỹ hơn về con trỏ trong chương về

con trỏ. Ở đây, chúng ta sẽ đề cập ngắn gọn đến hai toán tử được sử dụng để thao tác với các con trỏ.

Toán tử thứ nhất là &, là một toán tử quy ước trả về địa chỉ bộ nhớ của hệ số của nó.

Ví dụ:m = &count

Đặt vào biến m địa chỉ bộ nhớ của biến count.

Chẳng hạn, biến count ở vị trí bộ nhớ 2000, giả sử count có giá trị là 100. Sau câu lệnh trên m sẽ nhận giá trị 2000.

Toán tử thứ hai là *, là một bổ sung cho &; đây là một toán tử quy ước trả về giá trị của biến được cấp phát tại địa chỉ theo sau đó.

Ví dụ:q = *m

Sẽ đặt giá trị của count vào q. Bây giờ q sẽ có giá trị là 100 vì 100 được lưu trữ tại địa chỉ 2000.

Toán tử dấu phẩy ,

Toán tử dấu , được sử dụng để kết hợp các biểu thức lại với nhau. Bên trái của toán tử dấu , luôn được xem là kiểu void. Điều đó có nghĩa là biểu thức bên phải trở thành giá trị của tổng các biểu thức được phân cách bởi dấu phẩy.

Ví dụ:x = (y=3,y+1);

Trước hết gán 3 cho y rồi gán 4 cho x. Cặp dấu ngoặc đơn là cần thiết vì toán tử dấu , có độ ưu tiên thấp hơn toán tử gán.

Xem các dấu ngoặc đơn và cặp dấu ngoặc vuông là toán tử

Trong C, cặp dấu ngoặc đơn là toán tử để tăng độ ưu tiên của các biểu thức bên trong nó.

Các cặp dấu ngoặc vuông thực hiện thao tác truy xuất phần tử trong mảng.

Tổng kết về độ ưu tiên

<table>
<tr><td>Cao nhất</td><td>() []</td></tr>
<tr><td></td><td>! ~ ++ -- (Kiểu) * &</td></tr>
<tr><td></td><td>* / %</td></tr>
<tr><td></td><td>+ -</td></tr>
<tr><td></td><td><< >></td></tr>
<tr><td></td><td>< <= > >=</td></tr>
<tr><td></td><td>&</td></tr>
<tr><td></td><td>^</td></tr>
<tr><td></td><td>|</td></tr>
<tr><td></td><td>&&</td></tr>
<tr><td></td><td>||</td></tr>
<tr><td></td><td>?:</td></tr>
<tr><td></td><td></td></tr>
</table>

	= += -= *= /=
Thấp nhất	,

VI.2.9 Cách viết tắt trong C

Có nhiều phép gán khác nhau, đôi khi ta có thể sử dụng viết tắt trong C nữa. Chẳng hạn:

x = x + 10 được viết thànhx +=10

Toán tử += báo cho chương trình dịch biết để tăng giá trị của x lên 10.

Cách viết này làm việc trên tất cả các toán tử nhị phân (phép toán hai ngôi) của C. Tổng quát:

(Biến) = (Biến)(Toán tử)(Biểu thức)

có thể được viết:

(Biến)(Toán tử)=(Biểu thức)

CẤU TRÚC CỦA MỘT CHƯƠNG TRÌNH C

Tiền xử lý và biên dịch

Trong C, việc dịch (translation) một tập tin nguồn được tiến hành trên hai bước hoàn toàn độc lập với nhau:

- Tiền xử lý.

- Biên dịch.

Hai bước này trong phần lớn thời gian được nối tiếp với nhau một cách tự động theo cách thức mà ta có ấn tượng rằng nó đã được thực hiện như là một xử lý duy nhất. Nói chung, ta thường nói đến việc tồn tại của một

bộ tiền xử lý (preprocessor?) nhằm chỉ rõ chương trình thực hiện việc xử lý trước. Ngược lại, các thuật ngữ trình biên dịch hay sự biên dịch vẫn còn nhập nhằng bởi vì nó chỉ ra khi thì toàn bộ hai giai đoạn, khi thì lại là giai đoạn thứ hai.

Bước tiền xử lý tương ứng với việc cập nhật trong văn bản của chương trình nguồn, chủ yếu dựa trên việc diễn giải các mã lệnh rất đặc biệt gọi là các chỉ thị dẫn hướng của bộ tiền xử lý (destination directive of preprocessor); các chỉ thị này được nhận biết bởi chúng bắt đầu bằng ký hiệu (symbol) #.

Hai chỉ thị quan trọng nhất là:

- Chỉ thị sự gộp vào của các tập tin nguồn khác: #include

- Chỉ thị việc định nghĩa các macros hoặc ký hiệu: #define

Chỉ thị đầu tiên được sử dụng trước hết là nhằm gộp vào nội dung của các tập tin cần có (header file), không thể thiếu trong việc sử dụng một cách tốt nhất các hàm của thư viện chuẩn, phổ biến nhất là:

#include <stdio.h>

Chỉ thị thứ hai rất hay được sử dụng trong các tập tin thư viện (header file) đã được định nghĩa trước đó và thường được khai thác bởi các lập trình viên trong việc định nghĩa các ký hiệu như là:

#define NB_COUPS_MAX 100

#define SIZE 25

Cấu trúc một chương trình C

Một chương trình C bao gồm các phần như: Các chỉ thị tiền xử lý, khai báo biến ngoài, các hàm tự tạo, chương trình chính (hàm main).

Cấu trúc có thể như sau:

Các chỉ thị tiền xử lý (Preprocessor directives)#include <Tên tập tin thư viện>#define ….

Định nghĩa kiểu dữ liệu (phần này không bắt buộc): dùng để đặt tên lại cho một kiểu dữ liệu nào đó để gợi nhớ hay đặt 1 kiểu dữ liệu cho riêng mình dựa trên các kiểu dữ liệu đã có.Cú pháp: typedef <Tên kiểu cũ> <Tên kiểu mới>Ví dụ: typedef int SoNguyen; // Kiểu SoNguyen là kiểu int

Khai báo các prototype (tên hàm, các tham số, kiểu kết quả trả về,… của các hàm sẽ cài đặt trong phần sau, phần này không bắt buộc): phần này chỉ là các khai báo đầu hàm, không phải là phần định nghĩa hàm.

Khai báo các biến ngoài (các biến toàn cục) phần này không bắt buộc: phần này khai báo các biến toàn cục được sử dụng trong cả chương trình.

Chương trình chính phần này bắt buộc phải có<Kiểu dữ liệu trả về> main(){Các khai báo cục bộ trong hàm main: Các khai báo này chỉ tồn tại trong hàm mà thôi, có thể là khai báo biến hay khai báo kiểu.Các câu lệnh dùng để định nghĩa hàm mainreturn <kết quả trả về>; // Hàm phải trả về kết quả}

Cài đặt các hàm<Kiểu dữ liệu trả về> function1(các tham số){Các khai báo cục bộ trong hàm.Các câu lệnh dùng để định nghĩa hàmreturn <kết quả trả về>;}…

Lưu ý: Một số tập tin header thường dùng:

Một chương trình C bắt đầu thực thi từ hàm main (thông thường là từ câu lệnh đầu tiên đến câu lệnh cuối cùng).

Các tập tin thư viện thông dụng

Đây là các tập tin chứa các hàm thông dụng khi lập trinh C, muốn sử dụng các hàm trong các tập tin header này thì phải khai báo #include <Tên tập tin> ở phần đầu của chương trình

1) stdio.h: Tập tin định nghĩa các hàm vào/ra chuẩn (standard input/output). Gồm các hàm in dữ liệu (printf()), nhập giá trị cho biến (scanf()), nhận ký tự từ bàn phím (getc()), in ký tự ra màn hình (putc()), nhận một dãy ký tự từ bàm phím (gets()), in chuỗi ký tự ra màn hình (puts()), xóa vùng đệm bàn phím (fflush()), fopen(), fclose(), fread(), fwrite(), getchar(), putchar(), getw(), putw()…

2) conio.h : Tập tin định nghĩa các hàm vào ra trong chế độ DOS (DOS console). Gồm các hàm clrscr(), getch(), getche(), getpass(), cgets(), cputs(), putch(), clreol(),…

3) math.h: Tập tin định nghĩa các hàm tính toán gồm các hàm abs(), sqrt(), log(). log10(), sin(), cos(), tan(), acos(), asin(), atan(), pow(), exp(),…

4) alloc.h: Tập tin định nghĩa các hàm liên quan đến việc quản lý bộ nhớ. Gồm các hàm calloc(), realloc(), malloc(), free(), farmalloc(), farcalloc(), farfree(), …

5) io.h: Tập tin định nghĩa các hàm vào ra cấp thấp. Gồm các hàm open(), _open(), read(), _read(), close(), _close(), creat(), _creat(), creatnew(), eof(), filelength(), lock(),…

6) graphics.h: Tập tin định nghĩacác hàm liên quan đến đồ họa. Gồm initgraph(), line(), circle(), putpixel(), getpixel(), setcolor(), …

Còn nhiều tập tin khác nữa.

Cú pháp khai báo các phần bên trong môt chương trình C

Chỉ thị #include để sử dụng tập tin thư viện

Cú pháp:

```
#include <Tên tập tin> // Tên tập tin được đạt trong dấu <>
```

hay #include “Tên đường dẫn”

Menu Option của Turbo C có mục INCLUDE DIRECTORIES, mục này dùng để chỉ định các tập tin thư viện được lưu trữ trong thư mục nào.

Nếu ta dùng #include<Tên tập tin> thì Turbo C sẽ tìm tập tin thư viện trong thư mục đã được xác định trong INCLUDE DIRECTORIES.

Ví dụ: include <stdio.h>

Nếu ta dùng #include”Tên đường dẫn” thì ta phải chỉ rõ tên ở đâu, tên thư mục và tập tin thư viện.

Ví dụ:#include”C:\TC\math.h”

Trong trường hợp tập tin thư viện nằm trong thư mục hiện hành thì ta chỉ cần đưa tên tập tin thư viện. Ví dụ: #include”math.h”.

Ví dụ:

```
#include <stdio.h>

#include <conio.h>

#include “math.h”
```

Chỉ thị #define để định nghĩa hằng số

Cú pháp:

```
#define <Tên hằng> <Giá trị>
```

Ví dụ:

#define MAXINT 32767

Khai báo các prototype của hàm

Cú pháp:

<Kiểu kết quả trả về> Tên hàm (danh sách đối số)

Ví dụ:

long giaithua(int n); //Hàm tính giai thừa của số nguyên n

double x_mu_y(float x, float y);/*Hàm tính x mũ y*/

Cấu trúc của hàm "bình thường"

Cú pháp:

<Kiểu kết quả trả về> Tên hàm (các đối số)

{

Các khai báo và các câu lệnh định nghĩa hàm

return kết quả;

}

Ví dụ:

int tong(int x, int y) /*Hàm tính tổng 2 số nguyên*/

{

return (x+y);

```
}

float tong(float x, float y) /*Hàm tính tổng 2 số thực*/

{

return (x+y);

}
```

Cấu trúc của hàm main

Hàm main chính là chương trình chính, gồm các lệnh xử lý, các lời gọi các hàm khác.

Cú pháp:

```
<Kết quả trả về> main( đối số)

{

Các khai báo và các câu lệnh định nghĩa hàm

return <kết quả>;

}
```

Ví dụ 1:

```
int main()

{

printf(“Day la chuong trinh chinh”);

getch();

return 0;
```

```
}
```

Ví dụ 2:

```
int main()
{
int a=5, b=6,c;
float x=3.5, y=4.5,z;
printf("Day la chuong trinh chinh");
c=tong(a,b);
printf("\n Tong cua %d va %d la %d",a,b,c);
z=tong(x,y);
printf("\n Tong cua %f và %f là %f", x,y,z);
getch();
return 0;
}
```

BÀI TẬP

Bài 1: Biểu diễn các hằng số nguyên 2 byte sau đây dưới dạng số nhị phân, bát phân, thập lục phân

a)12b) 255c) 31000d) 32767e) -32768

Bài 2: Biểu diễn các hằng ký tự sau đây dưới dạng số nhị phân, bát phân.

a) ‘A’b) ’a’c) ‘Z’d) ’z’

Giới thiệu về ngôn ngữ C và môi trường turbo C 3.0
Học xong chương này, sinh viên sẽ nắm được các vấn đề sau: - Tổng quan về ngôn ngữ lập trình C. - Môi trường làm việc và cách sử dụng Turbo C 3.0.

TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH C

C là ngôn ngữ lập trình cấp cao, được sử dụng rất phổ biến để lập trình hệ thống cùng với Assembler và phát triển các ứng dụng.

Vào những năm cuối thập kỷ 60 đầu thập kỷ 70 của thế kỷ XX, Dennish Ritchie (làm việc tại phòng thí nghiệm Bell) đã phát triển ngôn ngữ lập trình C dựa trên ngôn ngữ BCPL (do Martin Richards đưa ra vào năm 1967) và ngôn ngữ B (do Ken Thompson phát triển từ ngôn ngữ BCPL vào năm 1970 khi viết hệ điều hành UNIX đầu tiên trên máy PDP-7) và được cài đặt lần đầu tiên trên hệ điều hành UNIX của máy DEC PDP-11.

Năm 1978, Dennish Ritchie và B.W Kernighan đã cho xuất bản quyển "Ngôn ngữ lập trình C" và được phổ biến rộng rãi đến nay.

Lúc ban đầu, C được thiết kế nhằm lập trình trong môi trường của hệ điều hành Unix nhằm mục đích hỗ trợ cho các công việc lập trình phức tạp. Nhưng về sau, với những nhu cầu phát triển ngày một tăng của công việc lập trình, C đã vượt qua khuôn khổ của phòng thí nghiệm Bell và nhanh chóng hội nhập vào thế giới lập trình để rồi các công ty lập trình sử dụng một cách rộng rãi. Sau đó, các công ty sản xuất phần mềm lần lượt đưa ra các phiên bản hỗ trợ cho việc lập trình bằng ngôn ngữ C và chuẩn ANSI C cũng được khai sinh từ đó.

Ngôn ngữ lập trình C là một ngôn ngữ lập trình hệ thống rất mạnh và rất "mềm dẻo", có một thư viện gồm rất nhiều các hàm (function) đã được tạo sẵn. Người lập trình có thể tận dụng các hàm này để giải quyết các bài toán mà không cần phải tạo mới. Hơn thế nữa, ngôn ngữ C hỗ trợ rất nhiều phép toán nên phù hợp cho việc giải quyết các bài toán kỹ thuật có nhiều công thức phức tạp. Ngoài ra, C cũng cho phép người lập trình tự định nghĩa thêm các kiểu dữ liệu trừu tượng khác. Tuy nhiên, điều mà

người mới vừa học lập trình C thường gặp “rắc rối” là “hơi khó hiểu” do sự “mềm dẻo” của C. Dù vậy, C được phổ biến khá rộng rãi và đã trở thành một công cụ lập trình khá mạnh, được sử dụng như là một ngôn ngữ lập trình chủ yếu trong việc xây dựng những phần mềm hiện nay.

Ngôn ngữ C có những đặc điểm cơ bản sau:

- Tính cô đọng (compact): C chỉ có 32 từ khóa chuẩn và 40 toán tử chuẩn, nhưng hầu hết đều được biểu diễn bằng những chuỗi ký tự ngắn gọn.
- Tính cấu trúc (structured): C có một tập hợp những chỉ thị của lập trình như cấu trúc lựa chọn, lặp… Từ đó các chương trình viết bằng C được tổ chức rõ ràng, dễ hiểu.
- Tính tương thích (compatible): C có bộ tiền xử lý và một thư viện chuẩn vô cùng phong phú nên khi chuyển từ máy tính này sang máy tính khác các chương trình viết bằng C vẫn hoàn toàn tương thích.
- Tính linh động (flexible): C là một ngôn ngữ rất uyển chuyển và cú pháp, chấp nhận nhiều cách thể hiện, có thể thu gọn kích thước của các mã lệnh làm chương trình chạy nhanh hơn.
- Biên dịch (compile): C cho phép biên dịch nhiều tập tin chương trình riêng rẽ thành các tập tin đối tượng (object) và liên kết (link) các đối tượng đó lại với nhau thành một chương trình có thể thực thi được (executable) thống nhất.

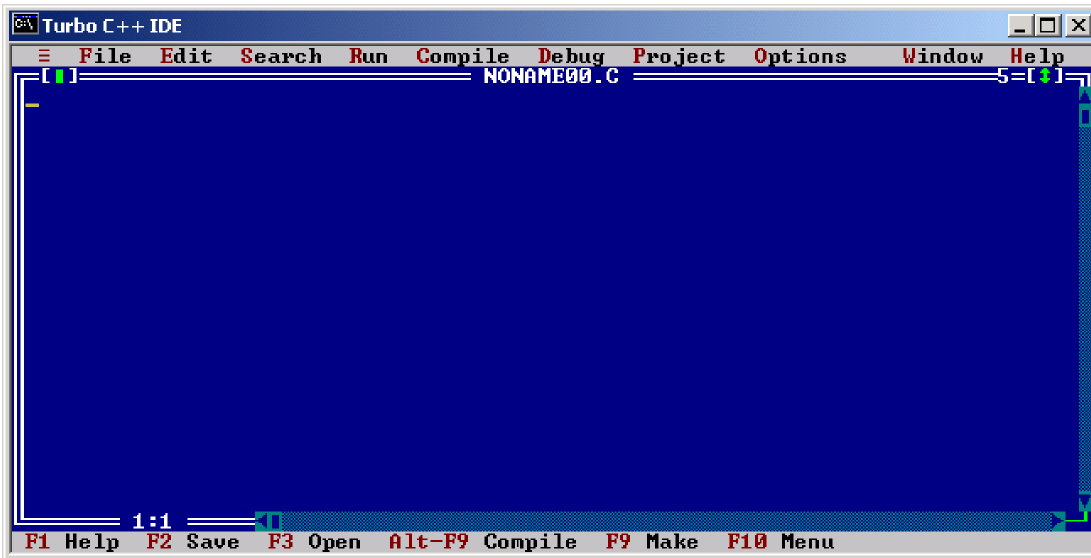
MÔI TRƯỜNG LẬP TRÌNH TURBO C

Turbo C là môi trường hỗ trợ lập trình C do hãng Borland cung cấp. Môi trường này cung cấp các chức năng như: soạn thảo chương trình, dịch, thực thi chương trình… Phiên bản được sử dụng ở đây là Turbo C 3.0.

Gọi Turbo C

Chạy Turbo C cũng giống như chạy các chương trình khác trong môi trường DOS hay Windows, màn hình sẽ xuất hiện menu của Turbo C có

dạng như sau:



Dòng trên cùng gọi là thanh menu (menu bar). Mỗi mục trên thanh menu lại có thể có nhiều mục con nằm trong một menu kéo xuống.

Dòng dưới cùng ghi chức năng của một số phím đặc biệt. Chẳng hạn khi gõ phím F1 thì ta có được một hệ thống trợ giúp mà ta có thể tham khảo nhiều thông tin bổ ích.

Muốn vào thanh menu ngang ta gõ phím F10. Sau đó dùng các phím mũi tên qua trái hoặc phải để di chuyển vùng sáng tới mục cần chọn rồi gõ phím Enter. Trong menu kéo xuống ta lại dùng các phím mũi tên lên xuống để di chuyển vùng sáng tới mục cần chọn rồi gõ Enter.

Ta cũng có thể chọn một mục trên thanh menu bằng cách giữ phím Alt và gõ vào một ký tự đại diện của mục đó (ký tự có màu sắc khác với các ký tự khác). Chẳng hạn để chọn mục File ta gõ Alt-F (F là ký tự đại diện của File)

Soạn thảo chương trình mới

Muốn soạn thảo một chương trình mới ta chọn mục New trong menu File (File ->New)

Trên màn hình sẽ xuất hiện một vùng trống để cho ta soạn thảo nội dung của chương trình. Trong quá trình soạn thảo chương trình ta có thể sử dụng các phím sau:

Các phím xem thông tin trợ giúp:

- F1: Xem toàn bộ thông tin trong phần trợ giúp.

- Ctrl-F1: Trợ giúp theo ngữ cảnh (tức là khi con trỏ đang ở trong một từ nào đo, chẳng hạn int mà bạn gõ phím Ctrl-F1 thì bạn sẽ có được các thông tin về kiểu dữ liệu int)

Các phím di chuyển con trỏ trong vùng soạn thảo chương trình:

Phím	Ý nghĩa	Phím tắt (tổ hợp phím)
Enter	Đưa con trỏ xuống dòng	
Mũi tên đi lên	Đưa con trỏ lên hàng trước	Ctrl-E
Mũi tên đi xuống	Đưa con trỏ xuống hàng sau	Ctrl-X
Mũi tên sang trái	Đưa con trỏ sang trái một ký tự	Ctrl-S
Mũi tên sang	Đưa con trỏ sang phải	Ctrl-D

phải	một ký tự	
End	Đưa con trỏ đến cuối dòng	
Home	Đưa con trỏ đến đầu dòng	
PgUp	Đưa con trỏ lên trang trước	Ctrl-R
PgDn	Đưa con trỏ xuống trang sau	Ctrl-C
	Đưa con trỏ sang từ bên trái	Ctrl-A
	Đưa con trỏ sang từ bên phải	Ctrl-F

Các phím xoá ký tự/ dòng:

Phím	Ý nghĩa	Phím tắt
Delete	Xoá ký tự tại ví trí con trỏ	Ctrl-G
BackSpace	Di chuyển sang trái đồng thời xoá ký tự đứng trước con trỏ	Ctrl-H

	Xoá một dòng chứa con trỏ	Ctrl-Y
	Xóa từ vị trí con trỏ đến cuối dòng	Ctrl-Q-Y
	Xóa ký tự bên phải con trỏ	Ctrl-T

Các phím chèn ký tự/ dòng:

Insert	Thay đổi viết xen hay viết chồng
Ctrl-N	Xen một dòng trống vào trước vị trí con trỏ

Sử dụng khối :

Khối là một đoạn văn bản chương trình hình chữ nhật được xác định bởi đầu khối là góc trên bên trái và cuối khối là góc dưới bên phải của hình chữ nhật. Khi một khối đã được xác định (trên màn hình khối có màu sắc khác chỗ bình thường) thì ta có thể chép khối, di chuyển khối, xoá khối... Sử dụng khối cho phép chúng ta soạn thảo chương trình một cách nhanh chóng. sau đây là các thao tác trên khối:

Phím tắt	Ý nghĩa

Ctrl-K-B	Đánh dấu đầu khối
Ctrl-K-K	Đánh dấu cuối khối
Ctrl-K-C	Chép khối vào sau vị trí con trỏ
Ctrl-K-V	Chuyển khối tới sau vị trí con trỏ
Ctrl-K-Y	Xoá khối
Ctrl-K-W	Ghi khối vào đĩa như một tập tin
Ctrl-K-R	Đọc khối (tập tin) từ đĩa vào sau vị trí con trỏ
Ctrl-K-H	Tắt/mở khối
Ctrl-K-T	Đánh dấu từ chứa chon trỏ
Ctrl-K-P	In một khối

Các phím, phím tắt thực hiện các thao tác khác:

Phím	Ý nghĩa	Phím tắt
F10	Kích hoạt menu chính	Ctrl-K-D, Ctrl-K-Q
F2	Lưu chương trình đang soạn vào đĩa	Ctrl-K-S
F3	Tạo tập tin mới	

Tab	Di chuyển con trỏ một khoảng đồng thời đẩy dòng văn bản	Ctrl-I
ESC	Hủy bỏ thao tác lệnh	Ctrl-U
	Đóng tập tin hiện tại	Alt-F3
	Hiện hộp thoại tìm kiếm	Ctrl-Q-F
	Hiện hộp thoại tìm kiếm và thay thế	Ctrl-Q-A
	Tìm kiếm tiếp tục	Ctrl-L

Ví dụ: Bạn hãy gõ đoạn chương trình sau:

```
#include <stdio.h>

#include<conio.h>

int main ()

{

char ten[50];

printf(“Xin cho biet ten cua ban !”);

scanf(“%s”,ten);

printf(“Xin chao ban %s”,ten);

getch();

return 0;

}
```

Ghi chương trình đang soạn thảo vào đĩa

Sử dụng File/Save hoặc gõ phím F2. Có hai trường hợp xảy ra:

- Nếu chương trình chưa được ghi lần nào thì một hội thoại sẽ xuất hiện cho phép bạn xác định tên tập tin (FileName). Tên tập tin phải tuân thủ quy cách đặt tên của DOS và không cần có phần mở rộng (sẽ tự động có phần mở rộng là .C hoặc .CPP sẽ nói thêm trong phần Option). Sau đó gõ phím Enter.

- Nếu chương trình đã được ghi một lần rồi thì nó sẽ ghi những thay đổi bổ sung lên tập tin chương trình cũ.

Chú ý: Để đề phòng mất điện trong khi soạn thảo chương trinh thỉnh thoảng bạn nên gõ phím F2.

Quy tắc đặt tên tập tin của DOS: Tên của tập tin gồm 2 phần: Phần tên và phần mở rộng.

- Phần tên của tập tin phải bắt đầu là 1 ký tự từ a..z (không phân biệt hoa thường), theo sau có thể là các ký tự từ a..z, các ký số từ 0..9 hay dấu gạch dưới (_), phần này dài tối đa là 8 ký tự.
- Phần mở rộng: phần này dài tối đa 3 ký tự.

Ví dụ: Ghi chương trình vừa soạn thảo trên lên đĩa với tên là CHAO.C

Thực hiện chương trình

Để thực hiện chương trình hãy dùng Ctrl-F9 (giữ phím Ctrl và gõ phím F9).

Ví dụ: Thực hiện chương trình vừa soạn thảo xong và quan sát trên màn hình để thấy kết quả của việc thực thi chương trình sau đó gõ phím bất kỳ để trở lại với Turbo.

Mở một chương trình đã có trên đĩa

Với một chương trình đã có trên đĩa, ta có thể mở nó ra để thực hiện hoặc sửa chữa bổ sung. Để mở một chương trình ta dùng File/Open hoặc gõ phím F3. Sau đó gõ tên tập tin vào hộp File Name hoặc lựa chọn tập tin trong danh sách các tập tin rồi gõ Enter.

Ví dụ: Mở tập tin CHAO.C sau đó bổ sung để có chương trình mới như sau:

```
#include <stdio.h>

#include<conio.h>

int main ()

{

char ten[50];

printf("Xin cho biet ten cua ban !");

scanf("%s",ten);

printf("Xin chao ban %s\n ",ten);

printf("Chao mung ban den voi Ngon ngu lap trinh C");

getch();

return 0;

}
```

Ghi lại chương trình này (F2) và cho thực hiện (Ctrl-F9). Hãy so sánh xem có gì khác trước?

Thoát khỏi Turbo C và trở về DOS (hay Windows)

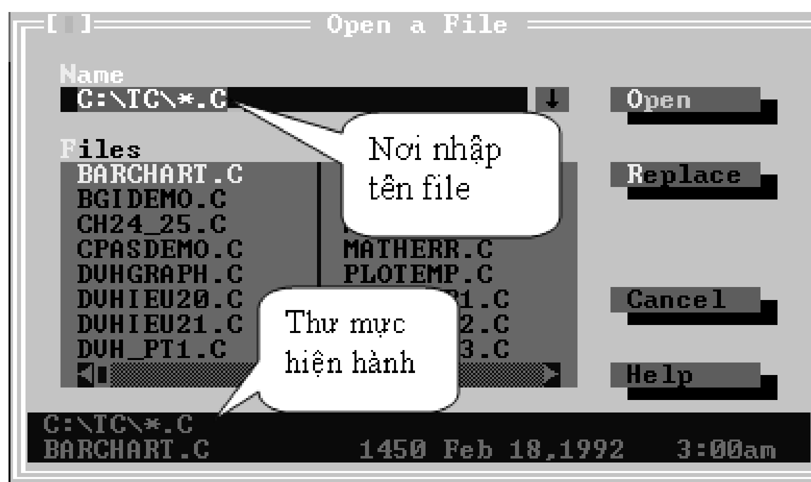
Dùng File/Exit hoặc Alt-X.

Sử dụng một số lệnh trên thanh menu

Các lệnh trên menu File (Alt -F)

SORRY, THIS MEDIA TYPE IS NOT SUPPORTED. - Lệnh New : Dùng để tạo mới một chương trình. Tên ngầm định của chương trình là NONAMEXX.C (XX là 2 số từ 00 đến 99).

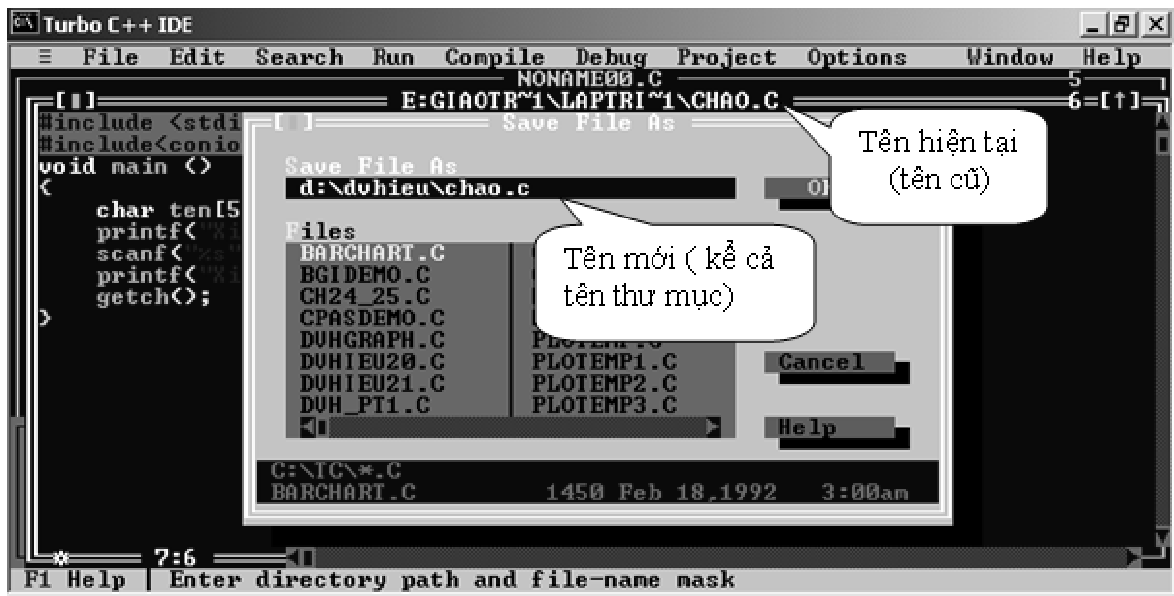
- Lệnh Open : Dùng để mở một chương trình đã có sẵn trên đĩa để sửa chữa, bổ sung hoặc để thực hiện chương trình đó. Khi tập tin được mở thì văn bản chương trình được trình bày trong vùng soạn thảo; hộp thoại Open như sau:



Trong trường hợp ta nhập vào tên tập tin chưa tồn tại thì chương trình được tạo mới và sau này khi ta lưu trữ, chương trình được lưu với tên đó.

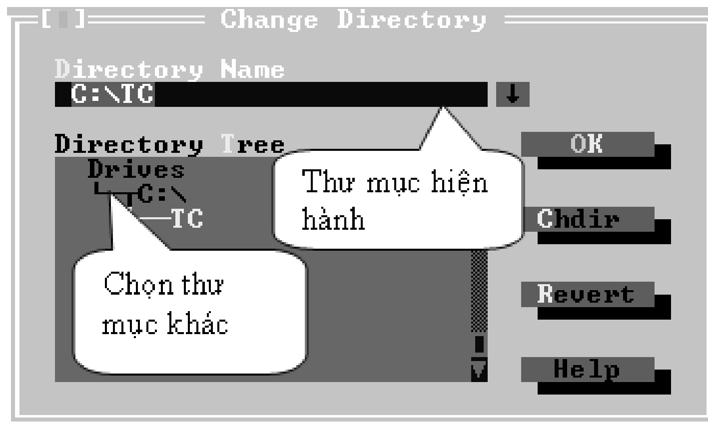
- Lệnh Save : Dùng để lưu chương trình đang soạn thảo vào đĩa.

- Lệnh Save as... : Dùng để lưu chương trình đang soạn thảo với tên khác, hộp thoại lưu tập tin đang soạn với tên khác như sau:



- Lệnh : Save All: Trong lúc làm việc với Turbo C, ta có thể mở một lúc nhiều chương trình để sửa chữa, bổ sung. Lệnh Save All dùng để lưu lại mọi thay đổi trên tất cả các chương trình đang mở ấy..

- Lệnh Change Dir ... : Dùng để đổi thư mục hiện hành



- Lệnh Print : Dùng để in chương trình đang soạn thảo ra máy in.

- Lệnh Printer Setup ...: Dùng để thiết đặt một số thông số cho máy in.

- Lệnh Dos Shell : Dùng để thoát tạm thời về Dos, để trở lại Turbo C ta đánh EXIT.

- Lệnh Exit : Dùng để thoát khỏi C.

Các lệnh trên menu Edit (Alt -E)

SORRY, THIS MEDIA TYPE IS NOT SUPPORTED. - Lệnh Undo : Dùng để hủy bỏ thao tác soạn thảo cuối cùng trên cửa số soạn thảo.

- Lệnh Redo : Dùng để phục hồi lại thao tác đã bị Undo cuối cùng.

- Lệnh Cut : Dùng để xóa một phần văn bản đã được đánh dấu khối, phần dữ liệu bị xóa sẽ được lưu vào một vùng nhớ đặc biệt gọi là Clipboard.

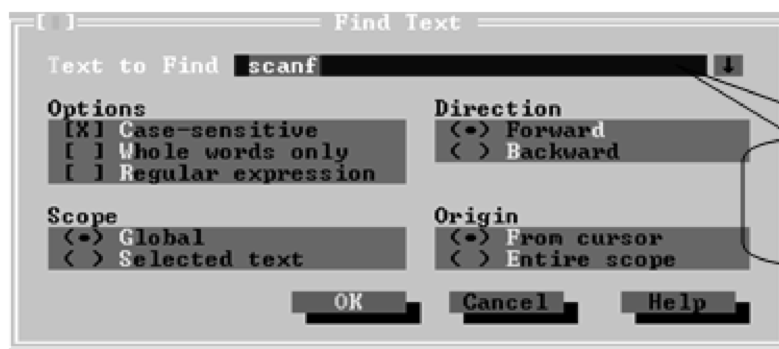
- Lệnh Copy : Dùng để chép phần chương trình đã được đánh dấu khối vào Clipboard.

- Lệnh Paste : Dùng để dán phần chương trình đang được lưu trong Clipboard vào của số đang soạn thảo, bắt đầu tại vị trí của con trỏ.

- Lệnh Clear : Dùng để xóa phần dữ liệu đã được đánh dấu khối, dữ liệu bị xóa không được lưu vào Clipboard.

SORRY, THIS MEDIA TYPE IS NOT SUPPORTED. - Lệnh Show clipboard : Dùng để hiển thị phần chương trình đang được lưu trong Clipboard trong một cửa sổ mới.

Các lệnh trên menu Search (Alt -S)

- Lệnh Find ...: Dùng để tìm kiếm một cụm từ trong văn bản chương trình. Nếu tìm thấy thì con trỏ sẽ di chuyển đến đoạn văn bản trùng với cụm từ cần tìm; hộp thoại Find như sau:
- 

Ý nghĩa các lựa chọn trong hộp thoại trên như sau:

- Case sentitive : Phân biệt chữ IN HOA với chữ in thường trong khi so sánh cụm từ cần tìm với văn bản chương trình.
- Whole word only: Một đoạn văn bản chương trình trùng với toàn bộ cụm từ cần tìm thì mới được xem là tìm thấy.
- Regular expression: Tìm theo biểu thức
- Global:Tìm trên tất cả tập tin.
- Forward : Tìm đến cuối tập tin.
- Selected text: Chỉ tìm trong khối văn bản đã được đánh dấu.
- Backward: Tìm đến đầu tập tin.
- From cursor : Bắt đầu từ vị trí con nháy.
- Entire scope: Bắt đầu tại vị trí đầu tiên của khối hoặc tập tin.

- Lệnh Replace...: Dùng để tìm kiếm một đoạn văn bản nào đó, và tự động thay bằng một đoạn văn bản khác, hộp thoại replace như sau:

Tìm các cụm từ Scanf và thay thế bằng scanf ***SORRY, THIS MEDIA TYPE IS NOT SUPPORTED.***

- Lệnh Search again : Dùng để thực hiện lại việc tìm kiếm.

- Các lệnh còn lại trên menu Search, các bạn sẽ tìm hiểu thêm khi thực hành trực tiếp trên máy tính.

SORRY, THIS MEDIA TYPE IS NOT SUPPORTED. Các lệnh trên menu Run (Alt -R)

- Lệnh Run : Dùng để thực thi hay "chạy" một chương trình.

- Lệnh Step over : Dùng để "chạy" chương trình từng bước.

- Lệnh Trace into : Dùng để chạy chương trình từng bước. Khác với lệnh Step over ở chỗ: Lệnh Step over không cho chúng ta xem từng bước "chạy" trong chương trình con, còn lệnh Trace into cho chúng ta xem từng bước trong chương trình con.

- Các lệnh còn lại, các bạn sẽ tìm hiểu thêm khi thực hành trên máy.

SORRY, THIS MEDIA TYPE IS NOT SUPPORTED. Các lệnh trên menu Compile (Alt C)

- Lệnh Complie: Biên dịch một chương trình.

- Lệnh Make , Build, … : Các lệnh này bạn sẽ tìm hiểu thêm khi thực hành trực tiếp trên máy tính.

- Lệnh Information : Dùng để hiện thị các thông tin về chương trình, Mode, môi trường .

SORRY, THIS MEDIA TYPE IS NOT SUPPORTED. Các lệnh trên menu Debug (Alt-D)

Trên menu Debug bao gồm một số lệnh giúp người lập trình "gỡ rối" chương trình . Người lập trình sử dụng chức năng "gỡ rối" khi gặp một số "lỗi" về thuật toán, sử dụng biến nhớ…

- Lệnh Breakpoints: Dùng để đặt "điểm dừng" trong chương trình. Khi chương trình thực thi đến "điểm dừng thì nó sẽ dừng lại" .

- Lệnh Watch : Dùng để mở một cửa sổ hiển thị kết quả trung gian của một biến nhớ nào đó khi chạy chương trình từng bước.

- Lệnh Evaluate/Modify: Bạn sẽ tìm hiểu khi thực hành trực tiếp trên máy.

SORRY, THIS MEDIA TYPE IS NOT SUPPORTED. Các lệnh trên menu Project (Alt- P)

Trên menu Project bao gồm các lệnh liên quan đến dự án như : đóng, mở, thêm , xóa các mục,…

SORRY, THIS MEDIA TYPE IS NOT SUPPORTED. Các lệnh trên menu Option (Alt -O)

Trên menu Option bao gồm các lệnh giúp người lập trình thiết đặt một số tự chọn khi chạy chương trình. Thông thường, người lập trình không cần phải thiết đặt lại các tự chọn.

- Lệnh Compiler ...: Dùng để thiết đặt lại một số thông số khi biên dịch chương trình như hình sau

Phần trình bày dưới đây thuộc về 3 mục: Directories, Enviroment và Save; các phần khác sinh viên tự tìm hiểu.

- Lệnh Directories...: Dùng để đặt lại đường dẫn tìm đến các tập tin cần thiết khi biên dịch chương trình như hình sau:

SORRY, THIS MEDIA TYPE IS NOT SUPPORTED.

- Include directory: Thư mục chứa các tập tin mà chúng ta muốn đưa vào chương trình (các tập tin .h trong dòng #include).
- Library directory : Thư mục chứa các tập tin thư viện (các tập tin .Lib)
- Output directory: Thư mục chứa các tập tin “đối tượng “ (có phần mở rộng là .OBJ), tập tin thực thi (.exe) khi biên dịch chương trình.
- Source directory: Thư mục chứa các tập tin “nguồn” (có phần mở rộng là .obj, .lib).

- Lệnh Environment: dùng để thiết lập môi trường làm việc như:

- Reference…: Các tham chiếu.
- Editor: Môi trường soạn thảo gồm: tạo tập tin dự phòng khi có sự chỉnh sửa (create backup file), chế độ viết đè (insert mode), tự động thụt đầu dòng (indent), đổi màu từ khóa (Syntax highlighting)… Đặc biệt, trong phần này là thiết lập phần mở rộng mặc định (Default Extension) của tập tin chương trình là C hay CPP (C Plus Plus: C++).

SORRY, THIS MEDIA TYPE IS NOT SUPPORTED.

- Mouse...: Đặt chuột.
- Colors…: Đặt màu.

Các lệnh trên menu Window (Alt- W)

Trên menu Window bao gồm các lệnh thao tác đến cửa sổ như:

- Lệnh Cascade : Dùng để sắp xếp các cửa sổ.

- Lệnh Close all : Dùng để đóng tất cả các cửa sổ.

- Lệnh Zoom: Dùng để phóng to/ thu nhỏ cửa sổ.

- Các lệnh Tile, Refresh display, Size/ Move, Next, Previous, Close, List...: Các bạn sẽ tìm hiểu thêm khi thực hành trực tiếp trên máy tính.

II.7.10. Các lệnh trên menu Help (Alt- H)

Trên menu Help bao gồm các lệnh gọi trợ giúp khi người lập trình cần giúp đỡ một số vấn đề nào đó như: Cú pháp câu lệnh, cách sử dụng các hàm có sẵn…

- Lệnh Contents: Hiện thị toàn bộ nội dung của phần help.

- Lệnh Index : Hiển thị bảng tìm kiếm theo chỉ mục.

- Các lệnh còn lại, bạn sẽ tìm hiểu khi thực hành trên máy.